Programs Using Multiple Files

Include Files

Organizing Programs

A large component of good programming is related to the organization of a program. Obviously, it is important for the programmer to understand their code.

As important, or perhaps more so, is making the code understandable to the next person who looks at it, such as another programmer (or the teacher who will be grading it!).

Organizing Programs

Some of the tools used to organize and help explain programs are:

- headers with information about the programmer and a description of the program
- functions & procedures declared at the top of programs
- meaningful names for variables and subprograms
- proper indentation & spacing

Size of Programs

As programs become more complex, it is inevitable that they will get longer. Large programs can easily stretch to hundreds, or even thousands, of lines of code.

In industry, programs containing over 1 million lines of code are quite common.

It is very difficult to focus on too much code at once, so we break it into pieces, or modules, to make it easier to understand.

Modularity with Subprograms

Functions and procedures are both examples of programming modularity. Code is removed from the main program and it is possible to focus on getting smaller pieces of code to work.

Even with good subprograms, the program files continue to get larger. A large file is slow to load, spans many screens, and is difficult to navigate.

Modularity with Multiple Files

Many programming languages also allow modularity by using multiple files. This reduces the size of individual files, making them easier to work with.

The main program must be in a single file. Thus it makes sense to use subprograms to simplify the main program, and keep functions and procedures in separate files to be used as needed.

Libraries of Subprograms

Most modern programming languages do some of this for you. For example, all of the string functions, such as length(), strint(), and intstr(), are part of the Turing string library.

There are many libraries with various functions and procedures for mathematics, formatting, graphics, file I/O, etc.

When you load Turing, they are automatically available.

Include Files

An include file is an external file that you must explicitly specify as part of your program. It usually contains subprograms, but could have other items such as mathematical constants (e.g., pi=3.14159)

Include files are sometimes referred to as header files.

In Turing, our include files should have the ".t" extension, just like the main program files.

Using Include Files

```
include "file1.t"
include "file2.t"

%%%%% Main Program %%%%%%%
...
do_this()  % from file1.t
...
do_that()  % from file2.t
...
```